Clase 4 — 18.11.25

#ERP

👨‍🏫 _Profesor: José Luis Sánchez Montejo
📘 Sistemas de gestión empresarial. ERP-CRM
🗓️ Clase 4 — 18/11/2025
🎯 Tema: Licencias y tipos de licenias (ERP, CRM, SRM, CMS, BI, Odoo...)

1️⃣7️⃣ Licencias de software en entornos empresariales (ERP, CRM, SRM, CMS, BI)

La elección de una licencia no es un asunto legal abstracto: determina hasta qué punto una empresa **controla**, **usa**, **modifica** o **redistribuye** el software que emplea en su infraestructura. Un ERP, un CRM o un CMS no son solo aplicaciones funcionales; son piezas críticas que gestionan procesos sensibles—clientes, facturación, inventario, web corporativa—y cuya licencia condiciona la arquitectura, el mantenimiento y la estrategia tecnológica futura.

Antes de evaluar un producto como Odoo, ERPNext o SuiteCRM, es imprescindible comprender si estás ante un software:

- completamente libre,
- libre pero con obligaciones fuertes,
- libre pero con pocas restricciones,
- cerrado pero gratuito (freeware),
- o directamente un producto de prueba limitado (shareware).

Cada categoría implica una relación distinta con la empresa y define si el software se convierte en un **activo estratégico** o en un **servicio alquilado**.

1️⃣8️⃣ Qué es una licencia y por qué importa

Una licencia es el **contrato legal** que define lo que está permitido hacer con un programa: usarlo, modificarlo, estudiarlo o redistribuirlo. Las funcionalidades del ERP o CRM importan, sí, pero están condicionadas por este marco legal.

Elegir un sistema sin entender su licencia puede generar dependencias, costes ocultos o incluso limitaciones legales al integrar módulos o realizar modificaciones internas.

Lo que determina una licencia

Una licencia establece:

- si el usuario puede **modificar el código**,
- si puede **redistribuirlo** o crear versiones derivadas,
- si debe usarlo exactamente tal cual lo distribuye el fabricante,
- si las modificaciones deben publicarse también como libres,
- si se transforma en un componente crítico propiedad tuya o si solo eres un inquilino que lo utiliza bajo permiso.

En un proyecto empresarial, esto influye en la autonomía técnica, en los costes de futuro y en la capacidad de evolución del sistema.

1️⃣9️⃣ GPL (General Public License)

La GPL es la licencia libre más influyente en la historia del software moderno. Su propósito no es solo permitir usar el código, sino **garantizar que la libertad se hereda** en cada modificación, distribución o derivada del software.
Esto se llama **copyleft fuerte**: si utilizas código GPL para crear un producto, ese producto hereda la misma obligación de ser GPL.

En el ecosistema empresarial esto tiene una consecuencia directa: **no puedes convertir un módulo basado en código GPL en un producto cerrado**, por más que lo hayas modificado o extendido.

La GPL se diseñó para evitar que empresas privatizaran mejoras sobre software comunitario.

Qué permite

La licencia es muy permisiva para el usuario final:

- uso personal, profesional y comercial sin restricciones,
- modificación total del código,
- redistribución del software original o modificado,
- acceso obligatorio al código fuente: no existen binarios sin su correspondiente código.

Qué obliga

Las obligaciones siempre aparecen **al redistribuir**:

- todo derivado debe liberarse también como GPL,
- si distribuyes binarios, debes entregar el código fuente completo,
- no puedes incorporar código GPL dentro de un software propietario sin liberar ese software completo bajo GPL.

En un ERP/CRM esto significa que un módulo que modifica directamente un componente GPL **no puede venderse como módulo privativo**.

Cuándo elegirla

La GPL se elige cuando:

- quieres proteger el ecosistema comunitario,
- quieres evitar la apropiación del código por terceros,
- deseas asegurar que todas las mejoras vuelvan a la comunidad.

Ejemplo clásico: **el kernel Linux**.

2️⃣0️⃣ AGPL (Affero GPL)

La AGPL nace para resolver un problema práctico: la GPL solo obliga a compartir cambios **si redistribuyes el software**, pero ¿qué pasa con los servicios web?

Un proveedor puede coger un software GPL, modificarlo y ofrecerlo como SaaS sin redistribuirlo nunca. Antes de AGPL, eso no obligaba a compartir modificaciones.

La AGPL corrige esto: **si ofreces el software modificado a usuarios finales a través de la red, también debes publicar los cambios**.

Es un copyleft fuerte aplicado al mundo de Internet.

Qué permite

- usar el software para crear servicios web y plataformas SaaS,
- combinarlo con infraestructura o servicios de cualquier tipo,
- utilizarlo comercialmente.

Qué obliga

- si alguien usa tu software **a través de la red**, tiene derecho a recibir el código fuente de la versión ejecutada,
- todas las modificaciones deben publicarse también bajo AGPL.

No importa si no redistribuyes binarios: basta con que los usuarios usen tu servicio por Internet.

Cuándo elegirla

- cuando quieres impedir que tu software libre se convierta en un SaaS cerrado,
- cuando buscas que las mejoras hechas por proveedores de servicios vuelvan a la comunidad,
- cuando el proyecto se basa en interacción web (CRMs, ERPs, plataformas colaborativas).

Ejemplo típico: **SuiteCRM**, basado en AGPL.

2️⃣1️⃣ LGPL (Lesser GPL)

La LGPL es un punto intermedio entre libertad y pragmatismo empresarial.
Su enfoque es claro: permitir que librerías libres convivan con software propietario sin “contaminarlo” con copyleft fuerte.

La diferencia clave es que la LGPL **solo impone obligaciones sobre la librería**, no sobre el programa final que la usa.

Qué permite

- enlazar (linking) librerías LGPL desde software propietario sin liberar el producto completo,
- incorporar componentes LGPL en aplicaciones comerciales sin compartir esas aplicaciones,
- uso comercial total.

Esto resuelve un problema crucial en empresas: pueden usar componentes libres **sin abrir su software interno**.

Qué obliga

- si modificas la librería LGPL, debes publicar esas modificaciones,
- debes permitir que el usuario pueda reemplazar la librería por otra versión (dinámicamente o mediante mecanismo equivalente),
- debes mantener la mención del autor y la licencia.

Cuándo elegirla

- cuando quieres fomentar adopción masiva de tu librería sin obligar a que el resto del software sea libre,

- cuando lo que quieres proteger es el núcleo técnico, pero no quieres imponer copyleft fuerte,
- cuando actúas como proveedor de componentes base (drivers, librerías gráficas, módulos de comunicación…).

Ejemplo: **las librerías estándar de GNU utilizadas por aplicaciones privativas sin abrir su código**.

2️⃣2️⃣ MIT

La licencia MIT es el símbolo de la filosofía **permisiva**: elimina barreras, maximiza la adopción y deja casi todo el control al usuario o a la empresa que integra el código.

No impone copyleft, no exige devolver cambios y permite relicenciar el software dentro de proyectos propietarios.
Es extremadamente corta y directa: “usa esto para lo que quieras, pero mantenme en los créditos”.

No tiene relación conceptual con la LGPL:

- MIT = libertarismo absoluto,
- LGPL = preserva la libertad de la librería, pero sin contaminar el resto.

Qué permite

- usar el código en productos privados y comerciales,
- modificarlo y no compartir tus modificaciones,
- relicenciarlo bajo otra licencia (incluyendo propietaria),
- integrarlo en frameworks, librerías o aplicaciones sin restricciones.

Qué obliga

Únicamente:

- mantener la nota de autoría original,
- eximir de responsabilidad al autor.

Nada más.

Cuándo elegirla

- cuando quieres que tu proyecto llegue al máximo número de desarrolladores y empresas,
- cuando no te importa que exista un ecosistema de versiones privativas basadas en tu código,
- cuando la prioridad es la adopción, no el control.

Ejemplo: **React (Facebook), TensorFlow, muchas librerías JS y Python**.

2️⃣3️⃣ Shareware (prueba limitada)

El shareware recuerda a los años 90, pero sigue presente en muchos productos comerciales modernos mediante sistemas de prueba.

Su modelo no se basa en libertad, sino en **convertir usuarios en clientes**: te dan una versión limitada del software y, si quieres más, pagas.

Características

- código cerrado,
- la redistribución está prohibida,

- no se permite modificar ni auditar,
- se ofrece acceso limitado por tiempo (30 días) o por funcionalidad (solo módulo básico),
- una vez caduca la prueba, hay que pagar.

Uso en entornos ERP/CRM

Suele aparecer en:

- versiones demo de ERPs propietarios,
- “trial editions” de CRMs comerciales,
- software de gestión orientado a pymes.

Es un modelo comercial, no una licencia libre.

2️⃣4️⃣ Freeware (no libre)

El freeware es una paradoja frecuente: “gratis” pero **no libre**.
Puedes usarlo, pero no puedes modificarlo, estudiarlo ni integrarlo profundamente.

El proveedor mantiene el control total: roadmap, soporte, funcionalidades y, sobre todo, el acceso al código.

Características

- es gratuito en precio, pero cerrado en permisos,
- útil para pequeñas empresas sin capacidades técnicas,
- no permite auditoría ni integraciones profundas,
- dependencia total del fabricante.

Ejemplo típico en CRM: **Zoho CRM Free Edition**, que ofrece funcionalidades básicas sin coste, pero todo el sistema es privativo.

2️⃣5️⃣ Tabla comparativa de licencias en ERP/CRM

Licencia	Filosofía	Obligaciones	Nivel de libertad	Ejemplos ERP/CRM
GPL	Copyleft fuerte	Derivadas deben ser GPL	Muy alta para el usuario, nula para privatizar	Odoo Community (antes de LGPL), Linux
AGPL	Copyleft fuerte aplicado a SaaS	Cambios deben liberarse aunque sea servicio web	Alta	SuiteCRM
LGPL	Copyleft débil	Liberar solo cambios en la librería	Alta y conciliadora	Odoo Community actual
MIT	Permisiva	Solo mantener autoría	Máxima	Librerías JS/Frameworks
Freeware	Gratuito cerrado	Uso limitado, sin modificación	Muy baja	Zoho Free
Shareware	Prueba limitada	Pago para versión completa	Muy baja	Trials de ERPs privativos

---

2️⃣6️⃣ Por qué no se puede copiar código Enterprise

Odoo Enterprise está licenciado como **software propietario**, lo cual implica:

- el código no es libre ni redistribuible,
- no puede copiarse ni reutilizarse en proyectos externos,
- no puede reverse-engineererarse para replicar funcionalidades,
- requiere licencia activa para cualquier uso comercial,
- no se permite modificarlo sin acuerdos formales.

Que puedas verlo en clase **no cambia su naturaleza legal**: se trata de un acceso pedagógico, igual que un profesor podría mostrar código de SAP para explicar cómo funciona un módulo financiero.

La prohibición de copiar no es una cuestión moral, sino contractual y jurídica: Odoo Enterprise está sujeto a un **contrato de licencia privativo** y a protección de propiedad intelectual.

Si quieres, puedo continuar ampliando **los tests de licencias**, o puedo enlazar esta parte con la siguiente sección de la clase: **modelos de negocio, dual licensing, caso real de Odoo Community vs Enterprise**, etc.# 2️⃣7️⃣ **Importancia de la licencia en decisiones empresariales**

La licencia condiciona aspectos estratégicos:

Coste a largo plazo

Un software libre puede implicar mayor inversión inicial en adaptación, pero elimina dependencias. Un software privativo puede parecer barato al inicio, pero convierte la continuidad del negocio en un contrato indefinido.

Integración técnica

Las licencias permisivas (MIT, BSD) permiten integrar sin restricciones. GPL obliga a planificar bien cómo se combinará código libre con módulos internos.

Escalabilidad futura

Un ERP libre puede evolucionar junto a la empresa. Un ERP privativo suele exigir pagar por módulos, usuarios, integraciones o mejoras.

Independencia del proveedor

Una licencia libre permite mantener tu propio sistema incluso si el proveedor desaparece o cambia condiciones comerciales.

2️⃣8️⃣ Tipos de software según la licencia

La clasificación de licencias no es un ejercicio teórico ni un catálogo legal. Es una **herramienta estratégica**. Determina cuánto control tiene una organización sobre sus sistemas informáticos —algo crítico en un ERP o CRM, donde el software forma parte directa del flujo de negocio.

Visualizar las licencias como un **espectro de libertad** permite entender rápidamente qué implica cada modelo:

- En el extremo izquierdo (GPL / AGPL) está el software que otorga al usuario el máximo poder: acceso total al código, derecho a modificarlo, libertad para auditarlo, replicarlo o migrarlo.

- En el extremo derecho (Freeware / Shareware) domina el proveedor: el usuario depende del soporte oficial, del roadmap impuesto, de las restricciones de uso y de la evolución comercial del fabricante.

El factor clave es **quién controla el futuro del producto**.
Cuando un ERP es crítico para cientos de procesos internos, perder ese control significa depender de terceros para cada cambio, parche o actualización.

En entornos donde la autonomía técnica es esencial, la elección de licencia puede determinar si un sistema crece con la empresa… o si la empresa queda atrapada en él.

2️⃣9️⃣ Tipos de software según la licencia

Hasta ahora solo se ha estudiado cada licencia individualmente, pero en un proyecto empresarial es necesario observar el **ecosistema completo**.

Un ERP no vive aislado: forma parte de una arquitectura que integra contabilidad, logística, producción, recursos humanos, reporting, firma digital, BI, etc.
Cada componente puede tener una licencia diferente y, dependiendo de ella, **la libertad del conjunto se amplía o se reduce**.

El tipo de licencia determina:

- si puedes extender el ERP internamente sin pagar una plataforma premium,
- si puedes integrar módulos externos sin restricciones,
- si puedes migrar datos y código sin romper contratos,
- si el proveedor puede cambiar condiciones unilateralmente (muy típico en SaaS privativo),
- si puedes mantener una versión antigua estable durante años.

En definitiva: **libertad del usuario ↔ control del proveedor**.

Libertad vs. control: un continuo

Al ordenar las licencias vistas en clase obtenemos un gradiente:

GPL → AGPL → LGPL → MIT/BSD → Freeware → Shareware

Cada paso hacia la derecha reduce libertades del usuario y aumenta la dependencia del proveedor.
Y lo importante: **no es un cambio técnico, sino de gobernanza**.

- En GPL/AGPL, el usuario controla el software.
- En LGPL/MIT, hay flexibilidad y colaboración sin obligaciones duras.
- En Freeware/Shareware, el usuario no controla nada más allá de lo que el fabricante permita.

Relevancia en ERPs y CRMs

Un ERP es, literalmente, el esqueleto digital de una empresa. Su licencia afecta:

- al coste total de propiedad (TCO),
- al tiempo necesario para implementar personalizaciones,
- a la capacidad de auditoría (clave en sectores regulados),
- a la seguridad (código disponible vs caja negra),
- al riesgo de **vendor lock-in**.

Por eso la clasificación no es un tema académico: es un **criterio de decisión** que condiciona todo el proyecto.

3️⃣0️⃣ Tres grandes categorías: Software libre, Open Source, Software propietario

La clase distingue tres grandes familias que suelen confundirse porque, a simple vista, las tres permiten "usar software". Pero sus implicaciones son radicalmente distintas:

1. **Software libre** → se centra en las libertades del usuario.
2. **Open Source (código abierto)** → se centra en la calidad, colaboración y eficiencia.
3. **Software propietario (privativo)** → se centra en el control del proveedor.

Comprender esta diferencia es clave antes de elegir un ERP o CRM.

Cuando una empresa decide un sistema de gestión, realmente está eligiendo:

- su **modelo de dependencia**,
- su **ritmo de innovación**,
- su **capacidad para adaptar procesos**,
- y su **libertad a largo plazo**.

En otras palabras: está eligiendo quién tiene el poder.

3️⃣1️⃣ Software libre

El Software Libre se basa en un principio esencial: **el control del software pertenece al usuario**.
No significa "software gratuito", sino "software que respeta las libertades fundamentales".

Las **4 libertades** son su núcleo:

1. **Usar el programa con cualquier fin**
   − desde un negocio autónomo hasta una multinacional.
2. **Estudiar cómo funciona**
   − implica acceso completo al código y documentación técnica.
3. **Modificarlo según tus necesidades**
   − adaptar procesos internos, añadir funcionalidades, integrar nuevos módulos.
4. **Distribuirlo**
   − compartir mejoras, colaborar con la comunidad o crear tu propia versión.

En un ERP, estas libertades permiten:

- corregir errores sin esperar a un proveedor,
- adaptar flujos de trabajo internos sin pagar licencias adicionales,
- integrar módulos externos o desarrollos propios,
- ejecutar versiones antiguas o personalizadas sin penalización contractual.

Ejemplos relevantes en clase

- **Odoo Community** (ERP modular libre con licencia LGPLv3)
- **GNU/Linux** (base de miles de servidores empresariales)
- **LibreOffice / GIMP** (alternativas libres frente a suites privativas)

Implicación real

El software libre convierte tu ERP en un **activo estratégico**, no en un alquiler condicionado.
Las organizaciones que buscan independencia tecnológica suelen preferirlo.

3️⃣2️⃣ Open Source (código abierto)

El Open Source comparte con el software libre el acceso al código, pero su origen y motivaciones son diferentes.
En los 90 surge como un movimiento que busca **industrializar el desarrollo colaborativo**, dejando de lado el componente filosófico.

Su objetivo no es liberar al usuario, sino mejorar la calidad técnica:

- cuanto más público es el código, más fácil es encontrar errores,
- grupos distribuídos pueden colaborar sin jerarquías rígidas,
- empresas pueden construir negocio sobre bases abiertas sin conflictos legales.

Características clave

- Se adopta por **eficiencia y calidad**, no por ideología.
- Permite modelos híbridos: parte del software abierta, parte privativa.
- Se adapta bien al mercado empresarial y al desarrollo en conjunto con comunidades.
- Permite modelos comerciales sólidos: soporte, consultoría, módulos premium…

En ERPs y CRMs esto es extremadamente común.

Ejemplos de clase

- **Android** → núcleo abierto, servicios Google cerrados.
- **VS Code** → base MIT, extensiones y Telemetry cerradas.
- **MySQL / MariaDB** → open source con versiones comerciales y soporte premium.
- **SuiteCRM (AGPL)**.

Relevancia para empresas

Muchas organizaciones no buscan un ERP “filosóficamente libre”, sino uno:

- abierto,
- auditable,
- extensible,
- con soporte comercial.

El Open Source encaja perfectamente en esa necesidad: ofrece un equilibrio entre libertad técnica y profesionalización.

3️⃣3️⃣ Software propietario o privativo

El software propietario ocupa el extremo más restrictivo del espectro. No se trata de “pagar por una copia del programa”, sino de **adquirir únicamente un permiso de uso**, condicionado, revocable y limitado por el fabricante. En otras palabras, el proveedor mantiene la propiedad del software en todo momento, y el usuario obtiene solo el derecho a ejecutarlo bajo unas reglas muy concretas.

Este modelo no concede acceso al código ni permite auditar cómo funciona internamente. Tampoco autoriza modificaciones; cualquier alteración del producto debe pasar por la empresa que lo desarrolla o por partners autorizados. Esto crea un ecosistema muy controlado y estable… a costa de la libertad del usuario.

Consecuencias clave del modelo privativo

1. **No se puede modificar el software**
   La empresa no tiene poder para adaptar internamente el ERP a sus procesos. Cada ajuste debe comprarse como servicio oficial o como desarrollo certificado.
2. **No se puede redistribuir**
   Incluso si se corrige un error o se implementa una mejora, no puede compartirse ni replicarse fuera de los límites contractuales.
3. **No hay acceso al código fuente**
   Esto impide:
   - auditorías de seguridad internas,
   - corrección de fallos críticos sin esperar al proveedor,
   - transparencia sobre cómo se gestionan los datos o las integraciones.
4. **Roadmap impuesto por el fabricante**
   El usuario no controla la evolución del software:
   - si el proveedor elimina un módulo, hay que adaptarse,
   - si cambia el precio, es obligatorio aceptarlo para continuar recibiendo soporte,
   - si modifica el modelo de licencias, afecta directamente al coste total de propiedad (TCO).

Este escenario genera lo que se conoce como **vendor lock-in**: dependencia estructural del proveedor. Migrar, cambiar de ERP o mantener versiones antiguas puede ser extremadamente costoso o directamente inviable.

Por qué sigue siendo atractivo para muchas empresas

A pesar de sus limitaciones, el software propietario ofrece ventajas importantes:

- **soporte oficial garantizado**,
- **actualizaciones regulares con planificación definida**,
- **ecosistema profesional certificado**,
- **menor riesgo para empresas sin equipo técnico interno**.

Una organización que prioriza previsibilidad, acompañamiento comercial y garantías contractuales puede preferir este modelo, aunque renuncie a la autonomía tecnológica.

Ejemplos en ERP/CRM

- **SAP S/4HANA** → control absoluto del proveedor, altísimo nivel de certificación.
- **Microsoft Dynamics 365** → ecosistema profesional y actualizaciones continuas.
- **Oracle ERP Cloud** → modelo cloud estricto con fuerte dependencia de servicio.
- **Salesforce CRM** → la plataforma CRM más extendida, totalmente privativa.

3️⃣4️⃣ Relación con Odoo y por qué esto importa

Este punto es clave para entender la arquitectura de Odoo y la lógica que guía tu aprendizaje.

Odoo no es un único producto: es un **ecosistema dual** dividido en dos ramas con filosofías totalmente diferentes.

Odoo Community → LGPLv3 (libre)

Esta versión:

- puede instalarse y usarse sin coste,
- permite modificar cualquier módulo,
- permite crear módulos propios (privativos o libres),
- puede integrarse con otros sistemas sin restricciones,
- puede mantenerse durante años sin depender del proveedor,
- otorga acceso completo al código.

Es la base sobre la que trabajan miles de empresas y desarrolladores.

Odoo Enterprise → licencia privativa (no libre)

Aquí la lógica cambia:

- el código no puede redistribuirse,
- cualquier copia fuera del contrato es una infracción,
- no se puede modificar internamente sin autorización,
- solo puede usarse mientras el contrato esté activo,
- incorpora módulos avanzados que no existen en Community.

Que el profesor muestre código Enterprise en clase no altera su naturaleza:
verlo no implica poder copiarlo, usarlo ni integrarlo fuera del entorno formativo.
Exactamente igual que ver código de SAP o Salesforce no te autoriza a reutilizarlo.

¿Por qué es crítico entender esto?

Porque esta distinción condiciona:

- la forma en que desarrollarás módulos,
- qué puedes publicar o no publicar,
- cómo diseñarás integraciones,
- qué partes puedes extender libremente,
- si puedes reutilizar código de ejemplo visto en clase.

Sin esta comprensión, podrías violar accidentalmente licencias sin saberlo.

3️⃣5️⃣ Implicaciones prácticas para un ERP/CRM real

Las licencias no solo definen libertades abstractas: **determinan cómo trabajará tu empresa a nivel técnico, económico y estratégico**. Aquí tienes un resumen con impacto real:

1. Desarrollo interno

Con **Odoo Community (LGPL)**:

- Puedes desarrollar módulos privados sin obligación de liberarlos.
- Solo debes abrir cambios si alteras la propia librería LGPL.
- Permite crear soluciones altamente personalizadas adaptadas a tus procesos internos.

Con **GPL/AGPL**:

- Si un módulo depende estrechamente del software GPL, debes liberar también tu derivado.
- En AGPL, incluso si solo se ofrece como SaaS, debes publicar los cambios.

2. Integraciones

Software libre (LGPL/MIT/GPL):

- puedes crear conectores propios, APIs internas y bridges sin restricciones,
- puedes abrir o cerrar el código del conector según tus necesidades,
- puedes adaptar el sistema a cualquier hardware o servicio.

Software propietario:

- solo se puede integrar mediante APIs oficiales,
- cualquier modificación que el proveedor no soporte queda prohibida,
- integraciones profundas requieren licencias o módulos premium.

3. Coste a largo plazo

Libre:

- no pagas licencias por usuario,
- pagas desarrollo, soporte, hosting y consultoría,
- el coste es estable y predecible, aunque depende del equipo técnico.

Privativo:

- coste ligado al número de usuarios,
- suscripciones anuales obligatorias,
- migraciones forzadas cuando el proveedor decide discontinuar versiones,
- costes variables y difícilmente negociables.

4. Estrategia tecnológica

Software libre:
El ERP crece contigo. Tú marcas el ritmo. Si necesitas un módulo crítico, puedes desarrollarlo sin pedir permiso. Puedes mantener versiones antiguas o personalizadas mientras tengan sentido.

Software propietario:
El futuro lo decide el proveedor. Tú te adaptas. La empresa pierde capacidad de maniobra cuando el negocio crece, cambia o evoluciona.

3️⃣6️⃣ Conclusión de la clase 4

Elegir un ERP no es una cuestión de modas, marcas o funciones. Es elegir un **modelo de gobernanza tecnológica**.

La licencia del software determina:

- **quién toma decisiones sobre el sistema**,
- **cómo se distribuyen los costes**,
- **si podrás crecer sin límites**,
- **si podrás mantener tu ERP 10 años**,
- **si dependerás de un proveedor o de tu propio equipo**,
- **si tus datos y procesos estarán en una caja negra o en un entorno auditable**.

Comprender GPL, AGPL, LGPL, MIT, Freeware o Shareware no es teoría jurídica:
es entender las reglas del juego que decidirán el éxito de cualquier proyecto ERP/CRM en el mundo real.

📝 Test — Preguntas, respuestas y justificación

✅ Pregunta 1

¿En qué caso la AGPL añade una obligación que no impone la GPL?

A. Cuando el software se usa solo dentro de una red local sin usuarios externos
B. Cuando el software modificado se ofrece como servicio web a usuarios por red
C. Cuando se compila con toolchains propietarios
D. Cuando se licencia documentación junto al código

Respuesta correcta: B

Justificación:
La AGPL añade una obligación adicional: si el software modificado se ofrece **como servicio accesible por red**, también deben liberarse los cambios. La GPL solo obliga al distribuir el programa, no al usarlo como servicio.

✅ Pregunta 2

Una empresa modifica un módulo de un ERP bajo GPL y lo distribuye a clientes. ¿Qué debe hacer?

A. Nada si cobra por la distribución
B. Publicar solo los parches críticos de seguridad
C. Liberar el código fuente de sus modificaciones bajo GPL
D. Enviar el binario bajo NDA sin código

Respuesta correcta: C

Justificación:
Si distribuyes un software derivado de código GPL, debes entregar también el **código fuente completo** de las modificaciones bajo GPL. Los binarios sin fuente o bajo NDA violan la licencia.

✅ Pregunta 3

¿Qué permite la LGPL que la GPL no permite en las mismas condiciones?

A. Redistribuir sin reconocer autoría
B. Enlazar una librería LGPL desde software propietario sin abrir todo el producto
C. Integrar el código en firmware sin atribución
D. Cerrar modificaciones a la librería LGPL

Respuesta correcta: B

Justificación:
La LGPL es “copyleft débil”: permite enlazar librerías LGPL dentro de software propietario sin obligar a liberar todo el programa. La GPL sí lo exigiría.

✅ Pregunta 4

Elige la opción que distingue correctamente Freeware de Shareware:

A. Freeware: abierto y gratuito; Shareware: abierto y de pago
B. Freeware: gratuito de uso, código cerrado; Shareware: prueba por tiempo/funciones y luego pago
C. Freeware: solo académico; Shareware: solo empresarial
D. Freeware: exige publicar cambios; Shareware: no

Respuesta correcta: B

Justificación:
El Freeware es gratis pero **código cerrado**. El Shareware es una **versión de prueba limitada** que requiere pagar para desbloquear funciones o continuar usándolo.

✅ Pregunta 5

¿Qué rasgo conceptual diferencia Software Libre y Open Source?

A. Open Source prioriza la colaboración/calidad técnica; el Software Libre prioriza la libertad del usuario
B. Open Source siempre exige copyleft fuerte
C. Software Libre permite cerrar derivadas; Open Source no
D. Open Source prohíbe el uso comercial, el Software Libre no

Respuesta correcta: A

Justificación:
El Software Libre nace con un enfoque **ético**: garantizar la libertad del usuario. El Open Source surge como enfoque **pragmático/empresarial** orientado a eficiencia, calidad y colaboración técnica.

✅ Pregunta 6

Si quieres máxima adopción para una librería UI que otros puedan integrar sin obligación de devolver cambios, ¿qué licencia elegirías?

A. GPL
B. AGPL
C. LGPL
D. MIT

Respuesta correcta: D

Justificación:
MIT es la licencia más **permisiva**: permite integrar la librería incluso en productos propietarios sin liberar mejoras. LGPL es permisiva, pero exige liberar modificaciones de la librería. GPL/AGPL generan obligaciones más fuertes.

✅ Pregunta 7

¿Cuál es la afirmación correcta?

A. Odoo Community es propietario y SAP es libre
B. Odoo Community se presenta como software libre; SAP S/4HANA es software propietario
C. Ambos son AGPL
D. Ambos son BSD

Respuesta correcta: B

Justificación:
Odoo Community está bajo licencia **LGPLv3**, por tanto es software libre. SAP S/4HANA es **software propietario**, de uso bajo licencia comercial.
